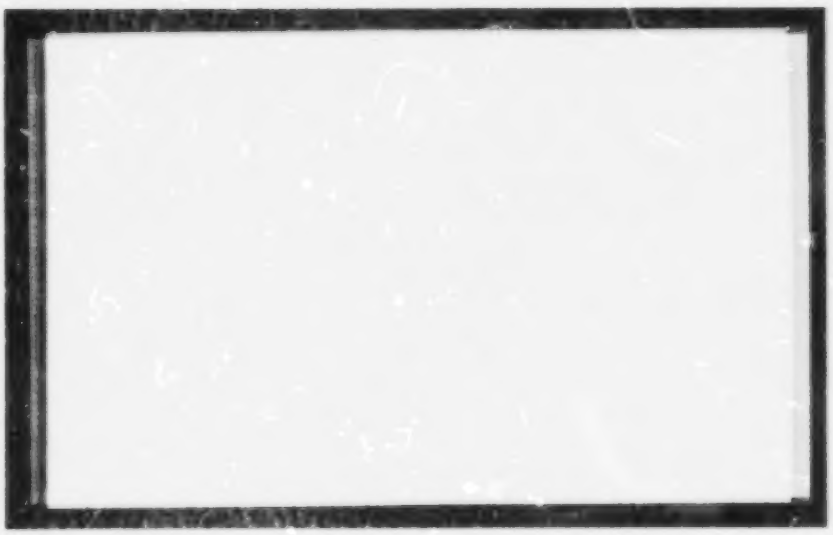
UNIVERSITÉ DE MONTRÉAL



CENTRE DE RECHERCHE SUR LES TRANSPORTS

CASE POSTALE 6128, SUCCURSALE A, MONTRÉAL, QUÉBEC, H3C 3J7

AN ACTIVE CONSTRAINTS NEWTON ALGORITHM FOR CONVEX COST MINIMIZATION WITH APPLICATION TO THE SPATIAL PRICE EQUILIBRIUM PROBLEM

by

Omar Drissi-Kaïtouni
and
Michael Florian

This research was supported in part by a grant form the Natural Sciences and Engineering Research Council of Canada.

Cette étude a été publiée grâce à une subvention du fonds F.C.A.R. pour l'aide et le soutien à la recherche.

Université de Montréal

Département d'Informatique et de
Recherche opérationnelle - Publication #617

Centre de recherche sur les transports - Publication #537

July 1987

Dépôt légal - 3e trimestre 1987

ABSTRACT

In this paper, we consider linearly constrained convex cost optimization problems with nonnegativity (or upper bound) constraints on the variables. For the solution of such problems, we propose a Newton type algorithm, where, at each iteration, an active constraint approach is used to satisfy the nonnegativity (or upper bound) constraints. The solution of the linearized subproblem in the Newton type algorithm is achieved by a method which we denote NAC. It solves linear systems of equations of the type By=b (B:nxn, y, b:nx1) by the use of a Gauss (G) or a Conjugate Gradient (CG) method. The CG method does not require the storing of the matrix B and permits the application of NAC-CG to problems of reasonably large size. This development was motivated by the search for an algorithm to solve General Spatial Price Equilibrium problems (GSPE), which we show to be equivalent to Network Equilibrium Problems (NEP). We tested the proposed algorithms on randomly generated problems and compared their performance to that of MINOS. The NAC-CG algorithm is the more efficient algorithm, based on our tests, for problems of reasonably large size (up to 700 variables).

Key words : Spatial Price Equilibrium on General Network, Network Equilibrium, Newton, Active Contraints, Conjugate Gradient.

RÉSUMÉ

Dans cet article, nous considérons les problèmes d'optimisation convexes avec contraintes d'égalité linéaires et contraintes de non négativité (nous admettons aussi des contraintes de capacité). Pour la résolution de ce type de problème, nous proposons une méthode de type Newton, où à chaque intération, une approche de type Contraintes Actives est utilisée pour tenir compte des contraintes de non négativité (et de capacité). L'étape importante de cet algorithme, que nous nommons NCA, concerne la résolution de Systèmes d'Equations Linéaires (SEL) du type By=b (B:nxn, y, b:nx1) et deux méthodes (Gauss (G) et Gradient Conjugué (GC)) sont envisagées. La méthode GC ne nécessitant pas le stockage de la matrice B, l'algorithme NCA-GC peut être utilisé pour les problèmes de grande taille.

Cette étude a été motivée par les Problèmes d'Équilibre Spatial sur Réseau Général (PESG) que l'on montre être des problèmes d'équilibre de réseau (PER), problèmes qui se formulent mathématiquement comme ceux pour lesquels NCA a été conçu. Plusieurs réseaux ont été générés aléatoirement afin de tester l'efficacité des algorithmes proposés. Les résultats numériques démontrent l'efficacité de NCA (Gauss ou GC) face au logiciel MINOS connu pour son efficacité. NCA avec la procédure GC s'avère le plus efficace, surtout pour les réseaux de grande taille.

Mots-clés : Équilibre Spatial sur Réseau Général, Équilibre de Réseau, Newton, Contraintes Actives, Gradient Conjugué.

INTRODUCTION

We present in this paper an active constraint Newton algorithm for the solution of convex cost minimization problems of the form

$$\min \quad f(x) \tag{1}$$

$$\text{subject to} \quad Ax=b \tag{2}$$

$$\text{and} \quad x>0 \tag{3}$$

where $x\varepsilon R^n$, A:mxn, m<n is a full rank matrix, $b\varepsilon R^m$ and $f:\Phi\rightarrow R$ is convex and twice differentiable on $\Phi=\{x\varepsilon R^n, Ax=b, x\geq 0\}$.

The motivation for the development of this algorithm is the solution of the spatial price equilibrium problem formulated in the space of arc flows, which we present in detail in a later section of this paper. Nevertheless, the method is sufficiently general to be applied to any convex cost minimization problem of the form (1)-(3) of moderate size.

The paper is organized as follows. In the next section, we consider the problem (1)-(2), that is without the nonnegativity contraints, and develop a Newton algorithm for its solution. Then, we reconsider the problem with the negativity constraints (3) and propose a particular active constraint algorithm for its solution, based on the results obtained for the unconstrained problem. Next, we give an interpretation of the algorithm in relation to classical active constraints methods; then we adapt the algorithm for the spatial price equilibrium problem and present computational results obtained with the adaptation of the algorithm for this problem.

1. THE NEWTON METHOD WITH EQUALITY CONSTRAINTS

We consider, in this section of the paper, the problem min $f(x)$, subject to $Ax=b$ ((1),(2)) and seek to solve it by a Newton method. This consists in solving a sequence of quadratic approximations of the problem. The general iteration of this algorithm, denoted by index k, consists then in solving the problem

$$\min \quad f(x^k) + \nabla f(x^k)^T(x-x^k) + \tfrac{1}{2}(x-x^k)\nabla^2 f(x^k)(x-x^k) \tag{4}$$

$$\text{subject to} \quad Ax=b \tag{5}$$

where $\nabla f(x^k)$ is the gradient of f evaluated at the current feasible solution x^k (which we will also denote as ∇f_k) and $\nabla^2 f(x^k)$ is the Hessian matrix of f evaluated at x^k (which we will also denote as D_k).

The Kuhn-Tucker optimality conditions for the problem (4), (5) are, for the solution (x^{k+1}, λ^k)

$$\nabla f(x^k) + D_k(x^{k+1} - x^k) + A^T\lambda^k = 0 \tag{6}$$

$$Ax^{k+1} = b \quad . \tag{7}$$

Let $d^k = (x^{k+1} - x^k)$ be the Newton direction of descent. (6) and (7) may be rewritten as

$$D_k d^k + A^T\lambda^k = -\nabla f(x^k) \tag{8}$$

$$Ad^k = b - Ax^k \quad . \tag{9}$$

We recall the following result stated as:

Lemma 1: (Luenberger (1984), p. 424). Consider two matrices A:mxn, m≤n, and D:nxn such that A is of full rank and D is positive definite. Then

$$H = \begin{bmatrix} D & A^T \\ A & 0 \end{bmatrix}$$

is a nonsingular matrix.

We use this result in the proof of

Theorem 1: Let $f \in C^2$, be a strictly convex function of $x \in E^n$ and A:mxn, m<n, a full rank matrix. Then $x^{k+1} = x^k + d^k$ is the optimal solution of (4)-(5) with λ^k, the vector of optimal multipliers associated to (5), where (d^k, λ^k) is the optimal solution of (8), (9).

Proof: By Lemma 1, the system of equations (8), (9) has a solution (d^k, λ^k). $(x^{k+1}, \lambda^k) = (x^k + d^k, \lambda^k)$, is then a Kuhn-Tucker point since it satisfies the equations (6,7) of (4,5). Since (4,5) is a convex cost minimization problem (x^{k+1}, λ^k) is its optimal solution. ∎

We further characterize the Newton algorithm with the following:

Theorem 2: Let $f \in C^2$, be a strictly convex function of $x \in E^n$ and A:mxn, m<n, be a full rank matrix. Let $x^{k-1} \in E^n$ and consider the Newton iteration (6), (7); then

1) x^k is feasible for (5);
2) if $d^k \neq 0$, d^k is a feasible descent direction for (4), (5);
3) if $d^k = 0$, then x^k is the optimal solution of (4), (5) with λ^k the optimal multiplier vector associated to (5).

Proof: i) Let x^k be the solution of (8), (9) for iteration k-1; then $Ad^{k-1} = b - Ax^{k-1}$, hence $Ax^k = A(x^{k-1} + d^{k-1}) = Ax^{k-1} + (b - Ax^{k-1}) = b$.

ii) Let x^k be feasible, that is $Ax^k = b$, by i); then (9) reduces to $Ad^k = b - Ax^k = 0$ and the system of equations (8), (9) is

$$D_k d^k + A^T \lambda^k = -\nabla f_k \qquad (8)$$

$$Ad^k = 0 \quad . \qquad (10)$$

If (8) is multiplied by d^k one obtains:

$$d^{k^T} D_k d^k + d^{k^T} A^T \lambda^k = -d^{k^T} \nabla f_k \quad .$$

Since $d^{k^T} A^T = 0$, it follows that (8) becomes

$$-d^{k^T} D_k d^k = d^{k^T} \nabla f(x^k) \tag{11}$$

If $d_k \neq 0$, then $d^{k^T} \nabla f(x^k) = -d^{k^T} D_k d^k < 0$ as D is a positive definite matrix and d^k is a descent direction of f. d^k is a feasible direction, since by (10), for a step size α

$$A(x^k + \alpha d^k) = Ax^k + 0 = b$$

since x^k is feasible. Thus d^k is a feasible direction for (1), (2). It remains now to consider the case $d^k = 0$.

iii) Equation (8) reduces to $D_k \cdot 0 + A^T \lambda^k = -\nabla f(x^k)$ which may be rewritten as $\nabla f(x^k) + A^T \lambda = 0$; since x^k is feasible, it follows that $Ax^k = b$. The point (x^k, λ^k) verifies the Kuhn-Tucker conditions:

$$\nabla f(x^*) + A^T \lambda^* = 0, \quad Ax^* = b \text{ , from (6), (7). } \blacksquare$$

Theorem 2 characterizes the Newton algorithm: the first iteration results in a feasible solution; for all subsequent iterations, if x^k is not the optimal solution it is possible to decrease the value of the objective function f since there exists α_k, $\alpha_k > 0$ such that $f(x^{k+1}) = f(x^k + \alpha_k d^k) < f(x^k)$, while all subsequent solutions are feasible since $A(x^{k+1}) = Ax^k + \alpha^k Ad^k = Ax^k = b$.

In the following section, we reconsider the nonnegativity constraints for the problem. The adaptation of the method for such contraints would rely, as much as possible, on the "good" properties of the Newton method presented above.

2. THE NEWTON METHOD WITH EQUALITY AND NONNEGATIVITY CONSTRAINTS

We consider in this section the problem (1)-(3) stated in the introduction, that is

$$\left.\begin{array}{rll} \min & f(x) & (1) \\ \text{subject to} & Ax=b & (2) \\ \text{and} & x\geq 0 & (3) \end{array}\right\} \quad (P)$$

In the following, we examine the conditions under which the Newton iteration of the previous section holds for (P) and how to avoid "blocking" of the direction by nonnegativity constraints which become binding. The Newton direction obtained by solving (8), (9) ensures that (2) is satisfied for $x^{k+1}= x^k+ \alpha_k d^k$, however (3) is not likely to be satisfied in general. It is relatively straightforward to characterize the feasibility of the Newton direction as stated by the following

Theorem 3: Let x^k be feasible for (P) and let d^k be the direction obtained by solving (8), (9). d^k is feasible if and only if for all i, $i\in\{1,\ldots,n\}$ such that $x_i^k= 0$, $d_i^k> 0$. (12)

Proof: The result of Theorem 2 ensures that $x^k+ \alpha_k d^k$ satisfies (3) for all α_k. d^k is feasible if and only if there exists an $\bar{\alpha}_k > 0$ such that $x^k+ \bar{\alpha}_k d^k> 0$. We consider two cases:

1: if $d_i^k= 0$, then $x_i^k+ \bar{\alpha}_k d_i^k= x_i^k> 0$ for all $\bar{\alpha}_k>0$;

2: if $d_i^k\neq 0$, then $x_i^k+ \bar{\alpha}_k d_i^k> 0$ if and only if

$$\bar{\alpha}_k \begin{cases} < -x_i^k/d_i^k \text{ if } d_i^k< 0 & (13) \\ > -x_i^k/d_i^k \text{ if } d_i^k> 0 & (14) \end{cases}$$

In (13) if $x_i^k > 0$ and $d_i^k < 0$ then $-x_i^k/d_i^k > 0$; if $x_i^k > 0$ then $x_i^k + \bar{\alpha}_k d_i^k > 0$ for $\bar{\alpha}_k \in [0, -x_i^k/d_i^k]$; if $x_i^k = 0$ then $-x_i^k/d_i^k = 0$ and hence there exists no $\bar{\alpha}_k > 0$ such that $x_i^k + \bar{\alpha}_k d_i^k > 0$. In (14) if $x_i^k > 0$ and $d_i^k > 0$ then $-x_i^k/d_i^k < 0$ and $x_i^k + \bar{\alpha}_k d_i^k > 0$ for any $\bar{\alpha}_k > \frac{-x_i^k}{d_i^k}$, hence for any $\bar{\alpha}_k > 0$. The only case then when the descent direction is blocked is where $x_i^k = 0$ and $d_i^k < 0$. ∎

The contrapose of Theorem 3 may be stated as

Corollary 1: If there exists at least one i, $i \in \{1, \ldots, n\}$ such that $x_i^k = 0$ and $d_i^k < 0$, then the Newton direction is not feasible for constraints (3).

It is worthwhile to note that by using (8) it is possible to identify the variables that block the Newton direction. If D_k is a diagonal positive definite matrix, $D_k = \text{diag}(D_i^k)$, $D_i^k > 0$ for all i then

$$d_i^k = -\frac{1}{D_i^k}\left(a_i^T \lambda^k + \nabla f_i(x^k)\right)$$

and if $d_i^k < 0$ then $\left(a_i^T \lambda^k + \nabla f_i(x^k)\right) > 0$.

At a given iteration k let λ^{k-1} be the solution of (8), (9) at the previous iteration. If the solution at iteration k is in the neighborhood of the optimal solution and the problem is not ill-conditioned, it is reasonable to suppose that λ^{k-1} is a good approximation of λ^k. If so, then the value of $\left(a_i^T \lambda^{k-1} + \nabla f_i(x^k)\right)$ is a good approximation of $\left(a_i^T \lambda^k + \nabla f_i(x^k)\right)$ and it is likely that d_i^k and $\left(a_i^T \lambda^{k-1} + \nabla f_i(x^k)\right)$ are of opposite sign more often than not. Based on this heuristic argument, let I and J be two sets of indices defined as follows:

$$I_k = \{ i \in A=\{1,\ldots,n\} : x_i^k = 0,\ g_i^k = (a_i^T \lambda^{k-1} + \nabla f_i(x^k)) > 0 \}$$

and $J_k = A - I_k$.

The variables corresponding to the indices in I_k are likely to be those that "block" the Newton direction and those corresponding to the indices in J_k are likely to be those for which either $x_j^k = 0$ and $d_j^k > 0$ or $x_j^k > 0$, thus rendering d_j^k, $j \in J_k$, a feasible direction of descent.

The algorithm that we suggest is based on the restrictions of the Newton algorithm to the elements of J_k. At each iteration the direction d^k is defined to be

$$d^k = \begin{cases} d_i^k = 0, & i \in I_k \\ d_j^k, & j \in J_k \end{cases} \qquad (15)$$

where $d_{J_k} = d_j^k$, $j \in J_k$ is the Newton direction obtained by solving the system (8), (9) restricted to the variables in the set J_k for which the nonnegativity constraints are relaxed.

As remarked before, the relaxation of these constraints does not ensure that the direction d_{J_k} is always feasible since the sets I_k, J_k are defined by employing the approximation of λ^k by λ^{k-1}.

For this reason, it is necessary to foresee that the algorithm can "unblock" itself if the direction computed by (15) is not feasible. In the same vein as an active constraint method, if blocking occurs, the sets I_k, J_k are modified as follows :

$$I_k \leftarrow I_k \cup \{ j : x_j^k = 0,\ d_j^k < 0 \}$$

$$J_k \leftarrow A / I_k$$

and the direction is recomputed as indicated in (15). By adopting this strategy, it is possible to include in the set I_k a sufficient number of variables in order to render the system

$$\begin{cases} Ad=0 \\ d_i=0, \ i\in I_k \end{cases}$$

singular, in which case the only possible solution for this system is d=0. In that case, one or more redundant constraints of the type $d_q=0$, $q\in I$ must be removed, which is equivalent to removing one index q of I at a time. In order to make the subsequent analysis precise, we state next the algorithm that we propose.

NCA

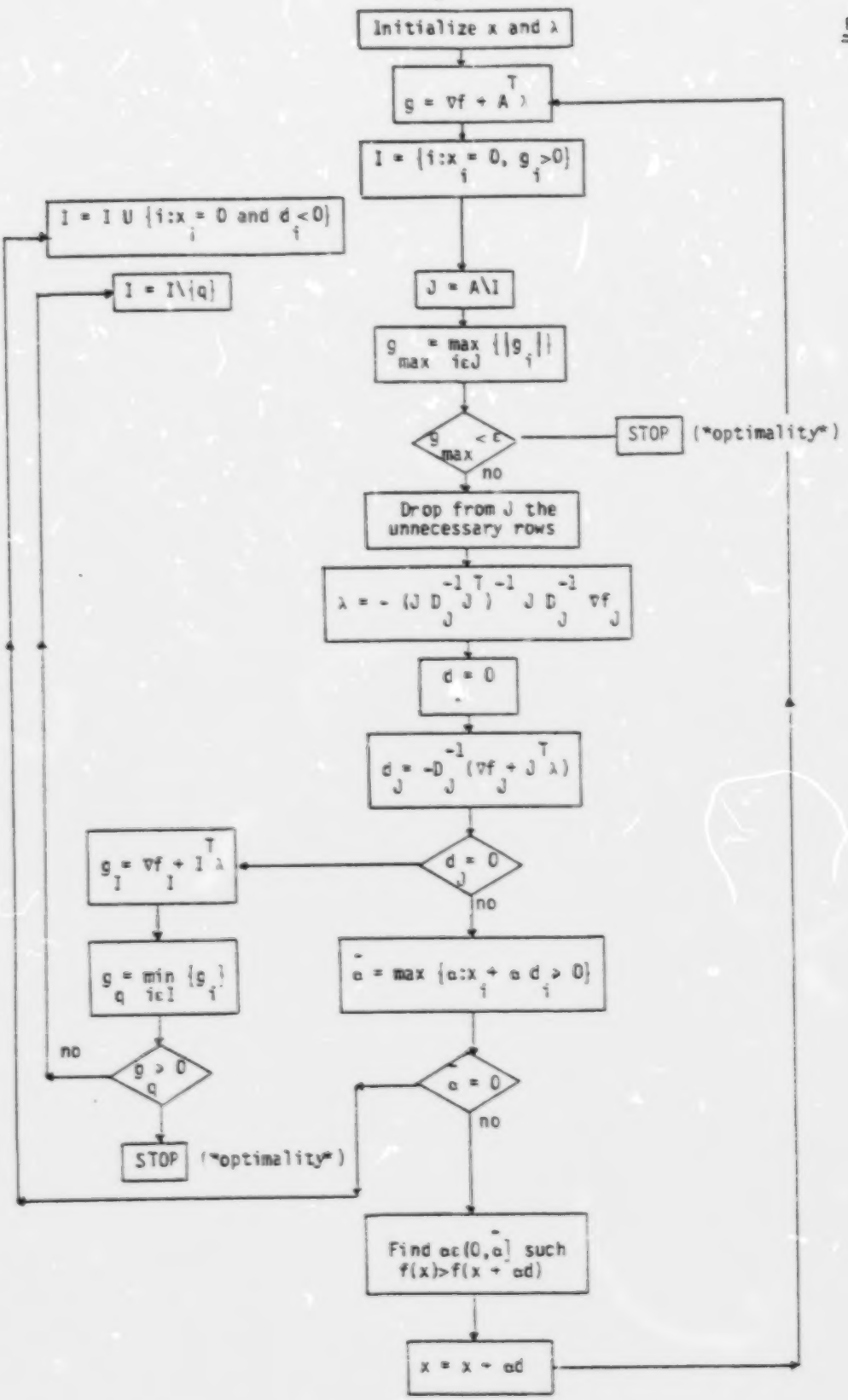

Initialize x and λ
g = ∇f + A^T λ
I = {i: x_i = 0, g_i > 0}
I = I ∪ {i: x_i = 0 and d_i < 0}
I = I\{q}
J = A\I
g_max = max_{i∈J} {|g_i|}
g_max < ε
STOP
("optimality")
no
Drop from J the unnecessary rows
λ = - (J D_J^{-1} J^T)^{-1} J D_J^{-1} ∇f_J
d = 0
d_J = -D_J^{-1} (∇f_J + J^T λ)
g_I = ∇f_I + I^T λ
d_J = 0
no
g_q = min_{i∈I} {g_i}
ᾱ = max {α: x_i + α d_i ≥ 0}
g_q > 0
no
α = 0
no
STOP
("optimality")
Find α∈(0, ᾱ] such f(x) > f(x + αd)
x = x + αd

Figure 2-1 : Block diagram of (NAC)

Algorithm (NAC):

0- Initialize $x \in \{x: Ax=b,\ x>0\}$, $\lambda=0$

1- $g = \nabla f + A^T\lambda$

2- $I = \{i:\ x_i = 0,\ g_i > 0\}$

3- $J = A \setminus I$

4- $g_{max} = \max_{i \in J} \{|g_i|\}$

5- If $g_{max} < \varepsilon$ then stop (ε-optimal solution attained)

6- Drop from J unnecessary rows

7- $\lambda = -(J\ D_J^{-1} J^T)^{-1}\ J\ D_J^{-1}\ \nabla f_J$

8- $d_I = 0$ and $d_J = -\ D_J^{-1}(\nabla f_J + J^T\lambda)$

9- If $d = 0$ go to 13-

10- $\bar{\alpha} = \max \{\alpha:\ x + \alpha d > 0\}$

11- If $\bar{\alpha} > 0$ go to 16-

12- $I = I \cup \{i:\ x_i = 0,\ d_i < 0\}$, and go to 3-

13- $g_I = \nabla f_I + I^T\lambda$ and define q such $g_q = \min_{i \in I} \{|g_i|\}$

14- If $g_q > 0$ then STOP (optimality of (P))

15- $I = I - \{q\}$ and go to 3-

16- $\alpha = \min\ (1, \bar{\alpha})$

17- If $f(x + \alpha d) < f(x)$ go to 19

18- $\alpha = \alpha/2$, and go to 17-

19- $x = x + \alpha d$ and go to 1-

It is interesting to note that in the Newton-Active-Constraints (NAC) algorithm the system

$$D_J\, d_J + J^T \lambda = -\nabla f_J$$

$$J\, d_J = 0$$

which is of dimension $(n_J + m_J) \times (n_J + m_J)$ is solved in the following way: the first equation is multiplied by $J\, D_J^{-1}$ to obtain

$$J\, D_J^{-1}\, D_J\, d_J + J\, D_J^{-1}\, J^T \lambda = -J\, D_J^{-1}\, \nabla f_J$$

$$\text{or} \quad J\, d_J + J\, D_J^{-1}\, J^T \lambda = -J\, D_J^{-1}\, \nabla f_J$$

and since $J\, d_J = 0$, λ is obtained by

$$\lambda = -(J\, D_J^{-1}\, J^T)^{-1}\, J\, D_J^{-1}\, \nabla f_J$$

$$\text{and } d_J \text{ by } d_J = -D_J^{-1}(\nabla f_J + J^T \lambda)\ .$$

The system of dimension $(n_J + m_J) \times (n_J + m_J)$ is reduced then to two systems of sizes $m_J \times m_J$ and $n_J \times n_J$. Furthermore, since D is diagonal so is D^{-1} and the computation of d_J becomes trivial, reducing the large system to a single system of size $m_J \times m_J$.

In the next section we give an interpretation of the algorithm and a convergence proof.

3. INTERPRETATION OF THE ALGORITHM AND PROOF OF CONVERGENCE

The algorithm that we propose is of the class of Newton methods where a typical iteration k may be generally stated as:

(NEWTON) i) Solve the quadratic approximation $q_k(y)$ of $f(x)$ at x^k

$$\min \tfrac{1}{2}(y-x^k)^T D_k (y-x^k) + \nabla f(x^k)^T (y-x^k) \qquad (16)$$

$$\text{subject to } Ay=b \qquad (17)$$

$$y \geq 0 \qquad (18)$$

we shall refer (16)-(18) as Q_k.

ii) Find a step size α^k for the direction $d^k = (y^k - x^k)$ which permits to decrease the value of the objective function, where y^k is the optimal solution of Q^k.

iii) $x^{k+1} = x^k + \alpha^k d^k$, $k=k+1$ (and return to i).

Step i) requires the solution of a quadratic programming problem, which may be solved by a variety of methods, including an active constraints (AC) method, such as described in Appendix 1. We have chosen to use a particular implementation of an AC method to solve Q_k. Also the quadratic subproblem Q_k is not necessarily solved to optimality; it suffices to find a solution y^ℓ such that $q_k(y^\ell) < q_k(y^0)$ (where $y^0 = x^k$). If such a feasible solution does not exist, then $y^0 = x^k$ is the optimal solution of Q_k.

In an AC method, a particular method must be chosen to initialize the set of active constraints. Our particular choice is to include the equality constraints (17) and the variables in the set I, for which $p_j = 0$. We denote the algorithm as a pseudo active constraints (PAC) algorithm which has the following steps:

(PAC) 0) Initialization: $\ell=0$, $y^0=x^k$, $W^0=E \cup I$
where E is the set of equality constraints, and I as defined above.

i) Solve $D_J\, p_J^\ell + J^T\lambda^\ell = -\nabla f_J^k$
$J\, p_J^\ell = 0$
if $p_J^\ell = 0$, go to iii)

ii) Let $\bar{\alpha} = \max\{\alpha : y^\ell + \alpha p^\ell > 0\}$
if $\bar{\alpha} > 0$, go to iv)
otherwise $I = I \cup \{i : y_i^\ell = 0,\ p_i^\ell < 0\}$ and go to i)

iii) Let $\mu_q = \min_{i \in I} \{\mu_i = (\nabla f_i^k + a_i^T\lambda^\ell)\}$ if $\mu_q > 0$, go to iv)
otherwise $I = I - \{q\}$ and go to i)

iv) STOP. Solution is optimal or $p_\ell \neq 0$.

The PAC algorithm corresponds to the NAC algorithm in steps 3 to 15: the initialization corresponds to step 2 of NAC; step i) corresponds to steps 3 to 8 of NAC, where steps 4 and 5 are an optimality test and step 6 is a test which ensures that the matrix $(J\, D_J^{-1}\, J^T)$ is nonsingular; step ii) corresponds to steps 10 to 12 of NAC and step iii) to steps 13 to 15. In step iv), if the optimal solution is not reached, one returns to the main problem. In step 16, if the optimal solution of the problem is not yet reached, a feasible descent solution has been obtained which serves to decrease the value of the objective function.

Thus, the NAC algorithm is a Newton type algorithm where the subproblem Q_k is solved by an active constraint type method. In the PAC algorithm used to solve Q_k, the choice of the set of active constraints is such that, in the neighborhood if the optimal solution is obtained in one iteration; furthermore, NAC behaves as a Newton algorithm in the neighborhood of the optimal solution subject only to equality constraints.

The PAC algorithm merits also the following observations. In step i) of the AC method (see Appendix 1) the problem to solve is

$$Qp^{\ell} + W^T J^{\ell} = -C - Qy^{\ell} \tag{19}$$

$$Wp^{\ell} = 0 \quad . \tag{20}$$

The objective function $q_k(y)$ of Q_k is

$$q_k(y) = \tfrac{1}{2}(y-x^k)^T D_k(y-x^k) + \nabla f_k^T(y-x^k)$$

$$= \tfrac{1}{2}y^T D_k y + (\nabla f_k - D_k x^k)^T y + \text{Constant} \ .$$

Hence, D_k corresponds to Q and $\nabla f_k - D_k x^k$ to C. We recall that in PAC $y^{\ell} = y^0$, $\ell > 0$ since once p^{ℓ} is nonzero PAC terminates; since $y^0 = x^k$, then $C + Qy^{\ell} = C + Qx^k$. Thus $C + Qy^{\ell} = C + Qx^k = (\nabla f_k - D_k x^k) + D_k x^k = \nabla f_k$. As a consequence (19), (20) may be rewritten

$$Qp^{\ell} + W^T J^{\ell} = -\nabla f_k \tag{21}$$

$$Wp^{\ell} = 0 \quad . \tag{22}$$

In (21), (22) Q=D and W consists of the contraints Ap=0 and the constraints $p_i = 0$, $i \in I$. Let λ be the dual variables that correspond to constraints Ap=0 and μ those that correspond to $-p_i = 0$, $i \in I$. Then δ^{ℓ} may be written as $\delta^{\ell T} = [\lambda^{\ell T} : -\mu^{\ell T}]$. (21), (22) may then be written as

$$D_k p^{\ell} + W^T \begin{bmatrix} \lambda^{\ell} \\ -\mu^{\ell} \end{bmatrix} = -\nabla f_k \tag{23}$$

$$Ap^{\ell} = 0 \tag{24}$$

$$-p_i^{\ell} = 0, \ i \in I \ . \tag{25}$$

Now, A may be split into I and J=A\J to obtain the system

$$D_J p_J + D_{JI} p_I + J^T\lambda = -\nabla f_J \tag{26}$$

$$D_{IJ} p_J + D_J p_I + I^T\lambda - \mu = -\nabla f_I \tag{27}$$

$$Jp_J + Ip_I = 0 \tag{28}$$

$$p_I = 0 \tag{29}$$

where the iteration indices k and ℓ have been omitted for clarity. Now by replacing (24) in (26)-(28) it follows that the system to be solved is

$$D_J p_J + J^T\lambda = -\nabla f_J \tag{30}$$

$$Jp_J = 0 \tag{31}$$

$$\mu = \nabla f_I + I^T\lambda + D_{IJ} p_J \tag{32}$$

$$p_I = 0 \tag{33}$$

One may recognize in (30), (31) the system solved in algorithm PAC with $p_I=0$ and, in step iii) of PAC $\mu_i = \nabla f_i + a_i^T\lambda$, $i\in I$ or $\mu = \nabla f_J + I^T\lambda$. However, in PAC step iii) may not be reached unless $p_J=0$ in i) and equation (32) with $p_J=0$ is verified in PAC.

Since it is unnecessary to dwell on the convergence of the method of active constraints (see Luenberger, (1984)) we end this section with the converge proof of the NAC algorithm.

Theorem 4: The Newton Active Constraint Algorithm generates a sequence $\{x^k\}$ from any feasible initial solution $x^0\in\Phi$ which converges to the optimal solution of the problem

$$\min\ f(x) \tag{P}$$

$$\text{subject to } Ax=b,\ x\geq 0$$

where $x \epsilon k^n$; A:mxn, m<n is a full rank matrix, $b \epsilon R^m$ and $f \epsilon C^2$ is strictly convex and attains its minimum on $\phi = \{x \epsilon R^n : Ax=b,\ x>0\}$.

Proof: We will show first that NAC is a descent algorithm. Let x^k be feasible and y^k the optimal solution of Q_k:

- if $y^k = x^k$ is the solution obtained by the active constraints algorithm, then x^k is the optimal solution of Q_k and hence of (P). This is straightforward to verify, as the Kuhn-Tucker conditions of (P) are

$$\nabla f(x^*) + A^T\lambda^* - \alpha^* = 0$$

$$Ax^* = b$$

$$x^* \geq 0,\ \alpha^* \geq 0,\ \alpha^{*T} x^* = 0$$

and for Q_k

$$\nabla f(x^k) + D_k(y^k - x^k) + A^T\lambda^k - \beta^k = 0$$

$$Ay^k = 0$$

$$y^k \geq 0,\ \beta^k \geq 0,\ y^{kT}\beta^k = 0 \ .$$

As $y^k = x^k$ then $D_k(y^k - x^k) = 0$ and $(x^k, \lambda^k, \beta^k)$ satisfy the optimality conditions of (P);

- if $y^k \neq x^k$ then, at some iteration ℓ in the subproblem Q_k the direction p^ℓ will be nonzero, as the solutions generated by the active constraints algorithm converge to y^k ($y^0 = x^k$ and $y^{\ell+1} = y^\ell + \alpha^\ell p^\ell$).

Then, the PAC algorithm terminates with $p^{\ell} \neq 0$ which satisfies

$$D_J p_J + J^T \lambda = -\nabla f_k \tag{34}$$

$$J p_J = 0 \tag{35}$$

$$p_I = 0 . \tag{36}$$

If (34) is multiplied by p_J one obtains

$$p_J^T D_J p_J + p_J^T J^T \lambda = -p_J^T \nabla f_J .$$

Since $p_J^T J^T \lambda = 0$ it follows that

$$p_J^T D_J p_J = -p_J^T \nabla f_J$$

and since $P_I = 0$ then

$$p_J^T D_J p_J = -p_J^T \nabla f_J - p_I^T \nabla f_J = -p^T \nabla f$$

then $p_J^T D_J p_J = -p^T \nabla f$; since $p^{\ell} = d^k$ then $-p_J^T D_J p_J = d^k \nabla f^k < 0$ as D_J is positive difinite. This proves that d^k is a descent direction for f; however p^{ℓ} is feasible for (2), (3) and so is d^k.

The rest of the proof is straightforward given the assumptions made on f. The algorithm generates a sequence $\{f(x^k)\}$ which is strictly decreasing and bounded from below by $f(x^*)$; this sequence converges and the continuity of f ensures that the sequence converges to $f(x^*)$. ∎

It is worthwhile to note that the NAC algorithm may be easily modified for upper bound constraints of the type $x<u$. Without giving all the details of the derivation, here are the two modifications necessary:

i) Choice of I: in the choice of the initial set of active constraints one includes the equality constraints E, the set I_0 of constraints $p_i=0$ for i such that $x_i^k=0$, $g_i^k + a_i^T \lambda^k > 0$ and the additional set I_u of constraints $p_i=0$ for i such that $x_i^k=u_i$ and $g_i^k + a_i^T \lambda^k < 0$. Thus $W = E \cup I_0 \cup I_u$.

ii) Feasibility of the descent direction: the direction of descent p always satisfies the equality constraints. In addition to the condition on $x>0$ ($x_i^k=0$ and $p_i^k< 0$) it is necessary to test the condition on $x<u$, that is $x_i^k=u_i$ and $p_i^k >0$.

We conclude this section with some general remarks on the NAC algorithm. It is evident that in the neighborhood of the optimal solution the subproblem Q_k will be solved to optimality and hence NAC will behave as a pure Newton method. Thus, in the neighborhood of the optimal solution the rate of convergence of the algorithm is quadratic. From a computational viewpoint the constraints $x>0$ ($x<u$) are advantageous in the sense that the system

$$D_J p_J + J^T\lambda = -\nabla f_J$$

$$J p_J = 0$$

is smaller than

$$Dp + A^T\lambda = -\nabla f$$

$$Ap = 0 \quad .$$

As remarked before, the solution of this (n+m)(n+m) system is carried out by solving two smaller systems of (nxn) and (mxm) which are $(AD^{-1}A^T)\lambda = -AD^{-1}\nabla f$ and $Dp= -\nabla f - A^T\lambda$. If D is diagonal the solution of the (nxn) system is trivial and one only has to solve an (mxm) system.

4. ADAPTATION OF THE NAC ALGORITHM FOR SOLVING SPATIAL PRICE EQUILIBRIUM PROBLEMS

The version of the spatial price equilibrium problem that we consider in this paper is its formulation in the space of the arc flows on a general network, such as formulated and studied by Friesz and Tobin (1981). As is well known, the descriptive equilibrium state of quantities bought, sold and transported on the links of a network may be obtained by solving the problem

$$\text{Min } F(d, v, b) = \sum_{\ell} \int_0^{d_\ell} S_\ell(x)dx + \sum_{a\in A} \int_0^{v_a} C_a(y)dy - \sum_{\ell} \int_0^{b_\ell} I_\ell(z)d_z \qquad (37)$$

$$\text{subject to} \qquad b_\ell + \sum_{a\in A_\ell} v_a - \sum_{a\in B_\ell} v_a - d_\ell = 0, \quad \forall \ell \qquad (38)$$

$$\text{and} \qquad d_\ell > 0,\ b_\ell > 0, \quad \forall \ell;\quad v_a > 0, \quad \forall a \qquad (39)$$

$S_\ell(\cdot)$ is the supply function at origin ℓ, $C_a(\cdot)$ is the average cost function for transportation of arc a and $I_\ell(\cdot)$ is the inverse demand function at destination ℓ. b_ℓ is the quantity supplied at origin ℓ and d_ℓ is the quantity bought at destination ℓ; v_a is the transportation flow on arc a. In order to state the conservation of flow equations A_ℓ is the set of arcs which have node ℓ as tail node and B_ℓ is the set of arcs which have node ℓ as head node.

This version of the problem is usually referred to as the General Spatial Price Equilibrium Problem (GSPEP) since it is formulated on a general network. Its formulation may be generalized to that of a single commodity network equilibrium problem (see Dafermos (1983), Drissi and Florian (1987)) by augmenting the original network by an additional node m, arcs (m,ℓ) for the origins and arcs (ℓ,m) for the destinations where

$$d_\ell = v_{m\ell};\ S_\ell(d_\ell) \doteq C_{m\ell}(v_{m\ell}) \tag{40}$$

$$b_\ell = v_{\ell m};\ I_\ell(b_\ell) \doteq -C_{\ell m}(v_{\ell m})$$

Let A^+ be the set of arcs in the augmented network. Then (37)-(39) may be reformulated as

$$\text{Min} \quad \sum_{a \in A^+} \int_0^{v_a} C_a(x)dx \tag{41}$$

$$\text{subject to} \quad \sum_{a \in A_\ell} v_a - \sum_{a \in B_\ell} v_a = 0, \quad \forall \ell \tag{42}$$

$$v_a > 0, \quad \forall a \tag{43}$$

where A_ℓ, B_ℓ are defined as before in the augmented network A^+.

When the right-hand side of equation (42) is nonzero and corresponds to demands from origins to destinations, (41)-(43) is a network equilibrium problem. In the application of the algorithm presented in the previous sections we do not impose that the right-hand side of (42) is zero. Thus, the computational approach is equally applicable to the GSPEP and to the one commodity network equilibrium problem (NEP). We assume that the functions $C_a(\cdot)$ are differentiable and monotone; hence the objective (41) is a strictly convex and twice differentiable function.

It is clear that this problem is a particular case of (1)-(3), with the exception that one of the equations in (42) is redundant. In order for (42) to be a full rank matrix, it suffices to remove one of the flow conservation equations, say the one for node m.

In the following, we give the adaptation of the algorithm presented in the preceding sections for both the GSPEP and the NEP. The only difference is in the determination of an initial solution, all the other steps of the algorithm are identical for both problems.

0 - Initialization:

For the GSPEP, v=0 is a feasible solution.

(For the NEP, an "all-or-nothing" assignment of the demands from origin to destination on shortest path is a feasible solution.)

λ=0 is a possible initial choice, but one may initialize λ using equation of step 7. Note that if the initial feasible solution v is $v > 0$, then the initial value of λ has no importance since the initial set I is empty.

1 - Computation of the gradient of the Lagrangean:

$$\nabla C + A^T\lambda = [C_a(v_a) - \lambda_i + \lambda_j]_{a \in A^+}, \quad \text{where } a=(i,j) .$$

λ_ℓ=0 for one of the node ℓ of the network, which is equivalent to removing one of the redundant constraints.

2-3-4-5 - Do not require comments.

6 - Rank of J and redundant constraints:

To test if J is of full rank, it suffices to test if the network, from which the arcs belonging to the set I have been removed, is connected. The following procedure is employed:

o) the arcs $a \in J$ are identified by p(a)=1 and
the arcs $a \notin J$ ($a \in I$) are identified by p(a)=0.
All nodes receive labels PT(i)=0; k=1.

i) label node k with PT(k)=2.

ii) for each a=(i,k) or (k,i) such that p(a)=1 do:
if PT(i)$\neq$2, then PT(i)=1.

iii) if PT(i)≠0 for all i, then J is of full rank. STOP.
Otherwise, if there exists i such that PT(i)=1, then k=i and return to i); otherwise J is not of full rank. STOP.

When this procedure terminates, the nodes i such that PT(i)=0 are not connected to the arcs in the set J, and the conservation of flow equations for these nodes are redundant. Thus, these lines ℓ are removed from J and λ_ℓ=0 for the current iteration for these nodes.

7 - Computation of λ:

$$\lambda = -(JD_J^{-1} J^T)\, JD_J^{-1} \nabla f_J \qquad \text{which may also be written as}$$

$$(JD_J^{-1} J^T)\lambda = -JD_J^{-1} \nabla f_J$$

which is a system of m equations and m variables.

There are two ways of solving this system: One is the use of pivoting method, such as the Gauss method (see Conte and DeBoor (1980)) and the other is its approximate solution by the conjugate gradient method. The advantage of the latter is that the matrix $(JD_J^{-1} J)$ need not be stored explicitly, which is advantageous for large scale problems. Moreover, as the solution is approximate, care must be taken to maintain the solution feasible, by using equation (9) rather than (10), so that if x^k is not feasible for Ax=b, then x^{k+1} is, by the result of theorem 2.

a) Computation of $JD_J^{-1} \nabla f_J$

$$D_J = \text{diag}(C'_a(v_a)) \text{ where } C'_a(v_a) = \frac{dC_a(v_a)}{dv_a} .$$

$$\text{Let } p = JD_J^{-1} \nabla f_J;\ p_i = \sum_{a \in A_i \cap J} C_a/C'_a - \sum_{b \in B_i \cap J} C_b/C'_b .$$

b) Solution of the system of equations for λ by the pivoting method. This method requires the matrix $(JD_J^{-1} J)$ which is easily computed by the following procedure:

o - M:=0

i - For each a=(i,j) do $M_{ii} := M_{ii} + 1/C'_a$

$M_{jj} := M_{jj} + 1/C'_a$

$M_{ij} := M_{ij} - 1/C'_a$

$M_{ji} := M_{ji} - 1/C'_a$.

In effect, the element (i,j) of the matrix $(JD_J^{-1} J^T)$ equals to $\sum_{a\in J} J_{ia}J_{ja}/C'_a$. However, for each a, $J_{ia}J_{ja} \neq 0$ if and only if $J_{ia}\neq 0$ and $J_{ja}\neq 0$. The only instances where $J_{ia} \cdot J_{ja} \neq 0$ are:

- i=j and a=(i,k) or a=(k,i); then $J_{ia} \cdot J_{ja} = (\pm 1)^2 = 1$;
- i≠j and a=(i,j) or a=(j,i); then $J_{ia} \cdot J_{ja} = \varepsilon(-\varepsilon)^{-1}$ with $\varepsilon=\pm 1$.

$(JD_J^{-1} J^T)$ is given by the matrix M computed by the above procedure. The algorithm used to solve the system $(JD_J^{-1} J^T)\lambda = -JD_J^{-1} \nabla f_J$ when $(JD_J^{-1} J^T)$ and $-JD_J^{-1} \nabla f_J$ are explicitly known is given in Appendix 2.

c) Solution of the system of equations for λ by the conjugate gradient method. The conjugate gradient algorithm used to solve systems of the type $M\lambda=b$ (M:mxm) is given in Appendix 3. It is important to note that this method only requires the explicit knowledge of vectors of the type p=My where y is known.

In our case, we need to compute $p=(JD_J^{-1} J^T)y$ without the explicit knowledge of the matrix $(JD_J^{-1} J^T)$. By an argumentation similar to the above, it is relatively straightforward to show that the vector p may be computed by the procedure:

o - p:=0

i - For each a=(i,j)∈J do:

$p_i : p_i + (y_i - y_j)/C'_a$

$p_j : p_j + (y_j - y_i)/C'_a$

8 - $d_I=0$ and computation of $d_J= -D_J^{-1} (\nabla f_J+ J^T\lambda)$:

For each $a=(i,j)\in J$ do $d_a= \frac{1}{C_a^+} (C_a- \lambda_i+ \lambda_j)$.

9-10-11-12 - Do not require comments.

13 - Computation of $\nabla f_I+ I^T\lambda$:

For each $a=(i,j)\in I$ do $g_a= C_a+ \lambda_j- \lambda_i$.

14-15-16-17 - Do not require comments.

18 - Computation of the step length:

Since we know that there exists $\alpha > 0$ such that $f(x+\alpha d) < f(x)$, by starting with $\alpha=\min(1, \bar{\alpha} > 0)$, the sequence $\{\alpha \leftarrow \frac{\alpha}{2}\}$ converges to zero and permits to find an $\alpha > 0$ which reduces the value of f. In effect, we use a bisection search.

19 - Does not require comments.

In the next section, we present computational results obtained with the algorithm.

5. NUMERICAL RESULTS

In the previous section, we considered a Newton-Active-Constraints algorithm to solve the GSPEP, and we proposed two approaches to solve the systems of linear equations (SLE). In the following, we call NACG the Newton-Active-Constraints algorithm where the SLE are solved by a Gauss method, and we denote NACCG the same algorithm where the SLE are solved by a conjugate gradient method. NACG and NACCG have been coded in FORTRAN and run on the CYBER 835 of the Université de Montréal. The termination criterion that has been used is that a norm of the reduced gradient, ∇_R, is less than a certain tolerance ε: $\|\nabla_R\|_\infty = \max_{j \in J} \{ g_j \} < \varepsilon = 10^{-4}$. Yet, we would like to note that in the numerical results obtained, the value of the norm of the reduced gradient is of the order of 10^{-6} to 10^{-12}; this is due to the quadratic rate of convergence of the method.

The aim of this section is to present the numerical results that demonstrate the efficiency of the proposed algorithms. In order to do so, we generated seven networks, that will be described later, and compared the numerical results obtained by NACG and NACCG on these problems with those obtained by MINOS (Murtagh and Saunders (1977)) which is well known for its efficiency. We note that Friesz and Tobin (1981) proposed a reduced gradient algorithm to solve the GSPEP and this method is the one used in MINOS. Thus, MINOS represents, to our knowledge, the state of the art in solving the GSPEP formulated in terms of arc flow variables.

The seven test networks have been randomly generated, and their dimensions are given in the first five columns of table 1. The number of variables of the problems is the number of arcs plus the number of destinations; thus, we used problems having between 42 and 691 variables. As it has been mentioned before, the size of the system of linear equations (SLE)

to be solved for a given problem is defined by the number of equality constraints, i.e. the number of nodes; thus, the size of SLE varies between 10x10 and 100x100.

Supply, demand and transportation cost functions are generated as follows:

at origins i: $C_{mi}(v_{mi}) = S_i(d_i) = \alpha_i + \beta_i d_i + \gamma_i d_i^2$ (44)

where $\alpha_i \in [0.1,1]$, $\beta_i \in [0.1,1]$ and $\gamma_i \in [0.01,0.05]$ are randomly generated from a uniform distribution.

at destinations j: $-C_{jm}(v_{jm}) = I_j(b_j) = \alpha_j/(1+b_j)$ (45)

where $\alpha_j \in [20,50]$ are randomly generated.

on the arcs a: $C_a(v_a) = \alpha_a + \beta_a v_a + \gamma_a v_a^2$ (46)

where $\alpha_a \in [0.1,1]$, $\beta_a \in [0.1,1]$ and $\gamma_a \in [0.5,2]$ are randomly generated.

We note that the transportation cost functions on the arcs a are quadratic, thus these problems are denoted by "Q" in table 1. For problems 1 to 4, we also used the following types of transportation cost functions:

- linear: $C_a(v_a) = \alpha_a + \beta_a v_a$ that are denoted by "L" in table 1,
- constant: $C_a(v_a) = \alpha_a$ that are denoted by "C" in table 1.

Before interpreting the numerical results in table 1, we would like to insist on the significance of the functions used. In the objective to be minimized (equation 37), the part relative to the origins is cubic (because of equation 45). Since most of the variables are arc flow variables (32/42 for network 1...), one can admit that the transportation cost functions of type (Q, L, or C) on the arcs, have a significant effect on the objective function.

Consider first the problems of type "C". In the objective function, the term $\sum_a \int C_a(x)dx$ is linear and its quadratic approximation is equal to its linear approximation. Thus, the Newton method reduces to a linear method (low convergence rate), and the lines "C" in table 1 reflect this fact for NAC and NACCG.

When we consider the problems "L", in the objective function, the term $\sum_a \int C_a(x)dx$ is quadratic, hence it is equal to its quadratic approximation. Therefore, we can expect that the Newton method will give the best results for this kind of problem, and lines "L" in table 1 show that this is true for NACG and NACCG.

In general, for nonlinear problems (L and Q) we see that NACG and NACCG have a better performance than MINOS in terms of the number of iterations (because of the quadratic rate of convergence) as well as in terms of computation time. On the other hand, we see that NACCG is more efficient (computation time) than NACG, particularly for large networks. This is due to the fact that the conjugate gradient method for solving systems of linear equations Ax=b does not need to store the matrix A, but only products of the form y=Ap.

From all the above, we conclude that the Newton Active Constraints method is very efficient to solve nonlinear GSPE problems, especially when the systems of linear equations are solved by a conjugate gradient method, as in NACCG algorithm. Since it does not require the storage of large matrices, NACCG is particularly useful for large networks; as shown in network 7, NACCG is approximately 6 times better than NAC and 22 times better than MINOS. Although a Gauss method may be used to efficiently solve systems Ax=b, NACCG appears to be the most efficient algorithm for small networks too.

Table 1: Numerical results relative to NACG, NACCG and MINOS

# NETWORK	# NODES	# OxD	# ARCS	# VAR	TYPE	NACG			MINOS		NACCG
						Niter	α=0	s.CPU	Niter	s.CPU	s.CPU
1	10	5x5	32	42	C	143	4	3.75	46	2.46	2.68
					L	8	4	0.48	55	2.97	0.55
					Q	9	6	0.51	51	2.48	0.50
2	20	10x10	117	137	C	118	7	9.75	94	8.72	14.29
					L	10	6	1.90	182	14.60	1.60
					Q	12	11	2.44	232	17.60	2.43
3	30	15x15	251	281	C	489	1	77.00	137	19.20	54.38
					L	10	7	4.53	369	48.30	3.36
					Q	17	12	7.09	434	54.30	5.75
4	40	20x20	440	480	C	219	11	68.80	340	57.10	128.34
					L	10	8	9.31	661	128.00	5.91
					Q	11	7	11.80	918	169.00	9.07
5	50	25x25	287	337	Q	18	23	20.40	714	105.00	10.03
6	50	25x25	549	599	Q	14	24	31.70	1144	248.00	14.46
7	100	50x50	591	691	Q	18	16	126.50	1632	473.00	22.38

CONCLUSION

In this paper, we consider convex nonlinear mathematical programs subject to linear equality constraints and nonnegativity constraints (there may also be capacity constraints). To solve this kind of problem, we develop a Newton type algorithm that we combine with an active constraints strategy to handle the nonnegativity constraints. At each iteration of this algorithm, which we call NAC, the choice of the initial active set is based on the value of the gradient and of the dual variables. Near optimality, this choice gives the optimal active set (corresponding to the actual value of dual variables), therefore NAC locally has a quadratic (Newton) convergence rate. From a computational viewpoint, we show that the active constraints $x=0$ (and $x=u$, if any), contribute to reduce the systems of linear equations (SLE) to be solved at each iteration. In this paper, we propose two ways of solving the SLE that are the Gauss method and the conjugate gradient method.

On the other hand, we are interested in solving General Spatial Price Equilibrium Problems (GSPEP) which consist in finding an equilibrium between production costs at origins plus transportation costs on the arcs of the network, and the market price at destinations. By adding one node and some arcs to the network, we show that the GSPEP is a particular Network Equilibrium Problem (NEP). When the cost function on each arc depends only on the flow on the same arc of the network, GSPEP and NEP may be written as nonlinear convex mathematical programs subject to linear equality and nonnegativity (perhaps also capacity) constraints, problems for which NAC has been designed.

The two versions of NAC (Gauss and Conjugate gradient to solve SLE) have been implemented on the CYBER 835 of the Université de Montréal. Several networks have been randomly generated to test the efficiency of the proposed algorithms. For comparison, we used the program MINOS (Martagh and Saunders (1977)) known for its efficiency.

Numerical results obtained demonstrate the efficiency of the Newton Active Constraints type algorithms mainly when the procedure of Conjugate gradient (CG) is used. Since in SLE problems of type Bx=b, the CG algorithm needs not to store the matrix B, NACCG may be used for large networks, and numerical results show that NACCG is particularly efficient for large problems; for instance, for a network with 100 nodes (SLE of size 100x100) and 691 variables, NACCG is approximately 22 times faster than MINOS.

We conclude that, for the problems that we tested, NACCG is the best algorithm to solve GSPEP and NEP and it is interesting as well for solving general linearly constrained convex cost minimization problems subject to nonnegativity (and upper bound) constraints in the variables.

APPENDIX 1

Active Constraints Method for Quadratic Programming

In the following, we summarize the Active Constraints method, described in Luenberger (1974) for quadratic programs of the form:

$$(P) \quad \min \tfrac{1}{2} y^T Qy + C^T y \tag{1}$$

$$\text{s.t.} \quad a_i^T y = b_i, \quad i \in E_1 \quad (\lambda_{E_1}) \tag{2}$$

$$a_i^T y \leq b_i, \quad i \in E_2 \quad (\lambda_{E_2}) \tag{3}$$

The following algorithm supposes that an initial feasible solution y^0 is known:

AC: 0 - Initialize y^0 feasible, and let $W^0 = E_1 \cup V$ where V is a subset of the set of indices $i \in E_2$ such that $a_i^T y^0 = b_i$. Let $\ell = 0$.

1 - Solve $$Qp^\ell + W^{\ell T} \lambda^\ell = -C - Qy^\ell \tag{4}$$

$$W^\ell p^\ell = 0 \tag{5}$$

where W^ℓ is the submatrix of A, composed of the lines in E_1 and V. If $p^\ell = 0$, go to 3.

2 - Let $y^{\ell+1} = y^\ell + \alpha^\ell p^\ell$ where $\alpha^\ell = \min(1, \bar{\alpha}^\ell)$

where $\bar{\alpha}^\ell = \max_{i \in E_2} \{\alpha : a_i^T (y^\ell + \alpha p^\ell) \leq b_i\}$.

If α^ℓ 1, let $W^{\ell+1} = W^\ell \cup \{q \in E_2 : a_q^T y^{\ell+1} = b_q\}$

otherwise $W^{\ell+1} = W^\ell$.

Go to 1.

APPENDIX 1 (cont'd)

3 - Let $\lambda_q = \min_{i \in V} \{\lambda_i\}$; if $\lambda_q > 0$ STOP, optimal solution found

otherwise $W^{\ell+1} = W^{\ell} - \{q\}$, $\ell = \ell+1$ and go to 1.

APPENDIX 2

Solution of Ax=b by a Gauss Method

A:nxn, x:nx1, b:nx1

0 - Initialize iord(i)=i, itest=invertible

1 - For i=1 to n, do:

i) $D_{iv}=A_{ii}$

ii) If $D_{iv}=0$, then: If ($A_{ik}=0$ for k=1 to n) then itest=noninvertable
stop
otherwise permute columns i and k where $A_{ik}\neq 0$
permute iord(i) and iord(k)

iii) For j=1 to n, do $A_{ij}= A_{ij}/D_{iv}$

iv) b_i/D_{iv}

v) For j=1 to n, do:
If j≠i and $A_{ji}\neq 0$, then Ratio = A_{ji}
For k=i to n, do: $A_{jk}= A_{jk}$ - Ratio * A_{ik}
$b_j= b_j$ - Ratio * b_i

2 - For i=1 to n, do: x(iord(i)) = b(i)

APPENDIX 3

Solution of $Hz=-g$ by the Conjugate Gradient Method

H:nxn, z:nx1, g:nx1

The following algorithm is drawn from Bertsekas (1982), where a sequence $\{z_m\}$ is generated:

$$z_0=0,\ z_{m+1}=z_m+\gamma_m p_m, \quad m=0,1,\ldots$$

where p_m is obtained by: $p_0=-Sr_0$, $p_m=-Sr_m+\beta_m p_{m-1}$, $m=1,2,\ldots$

The residual r_m is defined by: $r_m=Hz_m+g$, $m=0,1,\ldots$

and the scalars γ_m and β_m by:

$$\gamma_m=\frac{r_m^T Sr_m}{p_m^T Hp_m}, \quad m=0,1,\ldots$$

$$\beta_m=\frac{r_m^T Sr_m}{r_{m-1}^T Sr_{m-1}}, \quad m=1,2,\ldots$$

In this algorithm, the matrix S is a given definite positive matrix that one can take to be identity matrix, but a choice such $S=\mathrm{diag}(H_{ii})$ may accelerate convergence (Bertekas, 1982).

But the most important aspect to note in this algorithm is that the matrix H need not be stored explicitly; one needs only products of the form $y=Hv$ (Hz_m and Hp_m):nx1.

On the other hand, Bertsekas (1982) notes that only Hp_m has to be computed using H; Hz_m is obtained recursively by $Hz_{m+1}=Hz_m+\gamma_m Hp_m$ because $z_{m+1}=z_m+\gamma_m p_m$; and $Hz_0=0$ because $z_0=0$.

BIBLIOGRAPHY

BERTSEKAS, D., "Projected Newton Methods for Optimization Problems with Simple Constraints", SIAM J. Control and Optimization 20(2), 221-246 (1982).

BERTSEKAS, D., "Projected Methods for Variational Inequalities with Application to the Traffic Equilibrium Problem", Mathematical Programming Study 17, 139-159 (1982).

CONTE, S.D. and DE BOOR, C., Elementary Numerical Analysis, an Algorithmic Approach, third edition, Chap. 4, McGraw Hill (1980).

DAFERMOS, S. and NAGURNEY, A., "Isomorphism Between Spatial Price and Traffic Network Equilibrium Models", Lefschetz Center for Dynamical Systems, Brown University, Provident, RI LCDS #85-17 (1985).

DRISSI-KAÏTOUNI, O., "Un algorithme pour le Problème d'Équilibre Spatial Général exprimé dans l'espace des flots sur les chemins", publicationn #448, Centre de recherche sur les transports, Université de Montréal (1987).

DRISSI-KAÏTOUNI, O. "Problèmes d'équilibre spatial sur réseau général et méthodes de Newton", thèse de Ph.D., département d'Informatique et de Recherche opérationnelle, Université de Montréal (1987).

DRISSI-KAÏTOUNI, O. and FLORIAN, M., "An Algorithm for the Spatial Price Equilibrium Problem in Term of Path Flow Variables", publication #517, Centre de recherche sur les transports, Université de Montréal (1987).

FLORIAN, M., "On Some Generalizations of the Transportation Problem" Operational Research 81, P. Brans (ed.), IFORS, 607-620.

FLORIAN, M. and LOS, M., "A New Look at Static Spatial Price Models", Regional Science and Urban Economics 12, 579-597 (1982)

FRIESZ, T.L., and TOBIN, R.L., "Formulating and Solving the Spatial Equilibrium Problem with Transshipment in Terms of Arc Variables", Department of Civil and Urban Engineering, University of Pennsylvania (1981).

FRIESZ, T.L., HARKER, P.T., and TOBIN, L., "Alternative Algorithms for the General Network Spatial Price Equilibrium Problem", Journal of Regional Science 24(4), 475-507 (1984).

HITCHCOCK, F.L., "The Distribution of a Product from Several Sources to Numerous Localities", Journal of Mathematics and Physics 20, 224-230 (1941).

LUENBERGER, D.G., Linear and Nonlinear Programming, Chap. 6, 7, 11, Addison-Wesley (1984).

MURTAGH, B.A. and SAUNDERS, M.A., "MINOS User's Guide", technical report 77-9, Systems Optimization Laboratory, Stanford University, Stanford, CA (1977).

ORTEGA, J.M. and RHEINBOLDT, W.C., Iterative Solution of Nonlinear Equations in Several Variables, Chap. 13, Academic Press (1970).

SAMUELSON, P.A., "Spatial Price Equilibrium and Linear Programming", American Economics Review 42, 283-303 (1952).

SMITH, M.J., "The Existence, Uniqueness and Stability of Traffic Equilibria", Transportation Research 13B, 295-304 (1979).

TAKAYAMA, T. and JUDGE, G.C., "Spatial and Temporal Price and Allocation Models" North-Holland (1971).

WARDROP, J.G., "Some Theoretical Aspects of Road Traffic Research", Proceedings of the Institute of Civil Engineers, Part II 1, 325-378 (1952).

ZUBIETA, L., "Modèles d'Équilibre Spatial et méthodes de résolution", Ph.D. thesis, département d'Informatique et de Recherche opérationnelle, Université de Montréal (1987).